**Assignment(1)**

**Class:MCA-I(Sem-I)**

**Subject: Data Structures and Algorithms**

**Topics:**Array,need of array,Array representation,opeartions of array and difference between stack and queue

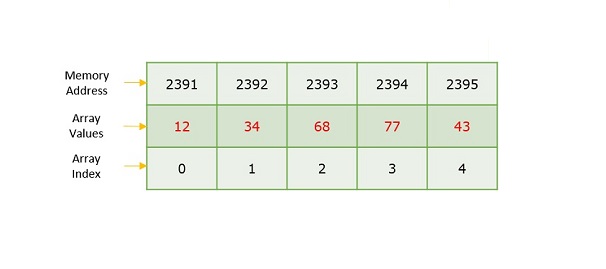
**Submitted To:**

Monika Bansal

Prof.of Comp.Sci.Department

**Ques1:What is Array?Need of Array ,representation of array and name the basic opeartions of array?**

**Ans**. Array is a type of linear data structure that is defined as a collection of elements with same or different data types. They exist in both single dimension and multiple dimensions. These data structures come into picture when there is a necessity to store multiple elements of similar nature together at one place.



The difference between an array index and a memory address is that the array index acts like a key value to label the elements in the array. However, a memory address is the starting address of free memory available.

Following are the important terms to understand the concept of Array.

* **Element** − Each item stored in an array is called an element.
* **Index** − Each location of an element in an array has a numerical index, which is used to identify the element.

Syntax

Creating an array in **C** and **C++** programming languages −

data\_type array\_name[array\_size] = {elements separated using commas}

or,

data\_type array\_name[array\_size];

Creating an array in **JAVA** programming language −

data\_type[] array\_name = {elements separated by commas}

or,

data\_type array\_name = new data\_type[array\_size];

Need for Arrays

Arrays are used as solutions to many problems from the small sorting problems to more complex problems like travelling salesperson problem. There are many data structures other than arrays that provide efficient time and space complexity for these problems, so what makes using arrays better? The answer lies in the random access lookup time.

Arrays provide **O(1)** random access lookup time. That means, accessing the 1st index of the array and the 1000th index of the array will both take the same time. This is due to the fact that array comes with a pointer and an offset value. The pointer points to the right location of the memory and the offset value shows how far to look in the said memory.

array\_name[index]

| |

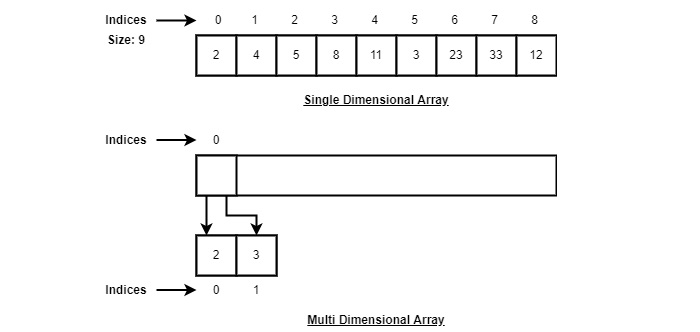
Pointer Offset

Therefore, in an array with 6 elements, to access the 1st element, array is pointed towards the 0th index. Similarly, to access the 6th element, array is pointed towards the 5th index.

Array Representation

Arrays are represented as a collection of buckets where each bucket stores one element. These buckets are indexed from ‘0’ to ‘n-1’, where n is the size of that particular array. For example, an array with size 10 will have buckets indexed from 0 to 9.

This indexing will be similar for the multidimensional arrays as well. If it is a 2-dimensional array, it will have sub-buckets in each bucket. Then it will be indexed as array\_name[m][n], where m and n are the sizes of each level in the array.



As per the above illustration, following are the important points to be considered.

* Index starts with 0.
* Array length is 9 which means it can store 9 elements.
* Each element can be accessed via its index. For example, we can fetch an element at index 6 as 23.

Basic Operations in the Arrays

The basic operations in the Arrays are insertion, deletion, searching, display, traverse, and update. These operations are usually performed to either modify the data in the array or to report the status of the array.

Following are the basic operations supported by an array.

* **Traverse** − print all the array elements one by one.
* **Insertion** − Adds an element at the given index.
* **Deletion** − Deletes an element at the given index.
* **Search** − Searches an element using the given index or by the value.
* **Update** − Updates an element at the given index.
* **Display** − Displays the contents of the array.

**Ques2:Difference between the Stack and Queue?**

**Ans.**

| **Stacks** | **Queues** |
| --- | --- |
| A stack is a data structure that stores a collection of elements, with operations to push (add) and pop (remove) elements from the top of the stack. | A queue is a data structure that stores a collection of elements, with operations to enqueue (add) elements at the back of the queue, and dequeue (remove) elements from the front of the queue. |
| Stacks are based on the LIFO principle, i.e., the element inserted at the last, is the first element to come out of the list. | Queues are based on the FIFO principle, i.e., the element inserted at the first, is the first element to come out of the list. |
| Stacks are often used for tasks that require backtracking, such as parsing expressions or implementing undo functionality. | Queues are often used for tasks that involve processing elements in a specific order, such as handling requests or scheduling tasks. |
| Insertion and deletion in stacks takes place only from one end of the list called the top. | Insertion and deletion in queues takes place from the opposite ends of the list. The insertion takes place at the rear of the list and the deletion takes place from the front of the list. |
| Insert operation is called push operation. | Insert operation is called enqueue operation. |
| Stacks are implemented using an array or linked list data structure. | Queues are implemented using an array or linked list data structure. |
| Delete operation is called pop operation. | Delete operation is called dequeue operation. |
| In stacks we maintain only one pointer to access the list, called the top, which always points to the last element present in the list. | In queues we maintain two pointers to access the list. The front pointer always points to the first element inserted in the list and is still present, and the rear pointer always points to the last inserted element. |
| Stack is used in solving problems works on [recursion](https://www.geeksforgeeks.org/introduction-to-recursion-data-structure-and-algorithm-tutorials/). | Queue is used in solving problems having sequential processing. |
| Stacks are often used for recursive algorithms or for maintaining a history of function calls. | Queues are often used in multithreaded applications, where tasks are added to a queue and executed by a pool of worker threads. |
| Stack does not have any types. | Queue is of three types – 1. Circular Queue 2. Priority queue 3. double-ended queue. |
| Can be considered as a vertical collection visual. | Can be considered as a horizontal collection visual. |
| Examples of stack-based languages include PostScript and Forth. | Examples of queue-based algorithms include Breadth-First Search (BFS) and printing a binary tree level-by-level. |